Using QIIME to Analyze 16S rRNA Gene Sequences from Microbial Communities

Justin Kuczynski[1], **Jesse Stombaugh**[2], **William Anton Walters**[1], **Antonio González**[3], **J. Gregory Caporaso**[4], and **Rob Knight**[2]

[1]Department of Molecular, Cellular, and Developmental Biology, University of Colorado, Boulder, Colorado

[2]Department of Chemistry and Biochemistry, University of Colorado, Boulder, Colorado

[3]Department of Computer Science, University of Colorado, Boulder, Colorado

[4]Department of Computer Science, Northern Arizona University, Flagstaff, Arizona

Abstract

QIIME (canonically pronounced "chime") is a software application that performs microbial community analysis. It is an acronym for Quantitative Insights In to Microbial Ecology, and has been used to analyze and interpret nucleic acid sequence data from fungal, viral, bacterial, and archaeal communities. The following protocols describe how to install QIIME on a single computer and use it to analyze microbial 16S sequence data from nine distinct microbial communities.

Keywords

microbial ecology; 16S; SSU; software; bioinformatics; QIIME

Introduction

A standard QIIME analysis begins with sequence data from one or more sequencing technologies, such as Sanger, Roche/454, Illumina, or others. Using QIIME to analyze data from microbial communities consists of typing a series of commands into a terminal window, and then viewing the graphical and textual output. Some fairly basic familiarity with a Linux-style command-line interface (i.e., the commands cd, ls, and the use of tab completion) is useful, though not required.

These protocols illustrate the use of QIIME to process data from a high-throughput 16S rRNA sequencing study, beginning with multiplexed sequence reads from a 454 sequencing instrument and finishing with taxonomic and phylogenetic profiles and comparisons of the samples in the study. Sequence data from Illumina and other platforms may be processed in a similar manner; see the Troubleshooting section of this unit for resources explaining the differences.

Rather than listing the analysis steps in general terms, we use an example of data from a study of the response of mouse gut microbial communities to fasting (Crawford et al., 2009). To make this example run quickly on a personal computer, we use a subset of the data generated from 5 animals kept on the control ad libitum fed diet, and 4 animals fasted for 24 hr before sacrifice. At the end of the basic protocols, we compare the community structure of control versus fasted animals, and, in particular, we compare taxonomic profiles of the microbial communities of both fasted and non-fasted mice, observe differences in diversity metrics within the samples and between the groups, and perform comparative clustering analysis to look for overall differences in the samples.

Basic Protocols 1-4 cover analysis of the mouse gut microbial communities described above. A Support Protocol covers installation of QIIME.

Basic Protocol 1

Acquiring an Example Study and Demultiplexing DNA Sequences

Basic Protocol 1 represents the first analysis steps typically performed on 16S DNA sequence data from microbial communities. Basic Protocol 1 consists of acquiring an example dataset, and assigning the DNA sequences in that study to the nine microbial communities included in the study. In typical usage, a researcher would substitute the data produced by a sequencing platform for the example data used here.

Necessary Resources—A functional installation of the QIIME VirtualBox is required; see Support Protocol for hardware requirements of the QIIME VirtualBox

1. Install the QIIME VirtualBox as described in Support Protocol, and start the VirtualBox.

2. First install a few files that are used when aligning 16S DNA sequences.

    Inside the QIIME VirtualBox, click on the black box with a > symbol on the top of the screen, which will open a terminal window (see Fig. 1E.5.1). Then, install the greengenes 16S alignment and lanemask, which will be used later to align sequences and filter out hypervariable regions.

    To do this, type:

    ```
    wget http://greengenes.lbl.gov/Download/Sequence_Data/
    Fasta_data_files/core_set_aligned.fasta.imputed
    ```

    (on a single line, with a space only after "wget"). Hit Enter, and then execute the following command:

    ```
    wget http://greengenes.lbl.gov/Download/Sequence_Data/
    lanemask_in_1s_and_0s
    ```

3. Next acquire data from an example experiment. In the terminal window, type:

    ```
    wget http://bmf.colorado.edu/QIIME/qiime_tutorial-v1.3.0.zip

    unzip qiime_tutorial-v1.3.0.zip
    ```

```
cd qiime_tutorial-v1.3.0
```

The files present in this directory are examples provided by the QIIME developers; they include the following.

Sequences (`.fna`): This is the 454-machine-generated FASTA file. Using the Amplicon processing software on the 454 FLX standard, each region of the PTP plate will yield a FASTA file of form `1.TCA.454Reads.fna`, where `1` is replaced with the appropriate region number. For the purposes of this tutorial, we use the FASTA file `Fasting_Example.fna`.

Quality Scores (`.qual`): This is the 454-machine-generated quality score file, which contains a score for each base in each sequence included in the FASTA file. Like the FASTA file mentioned above, the Amplicon processing software will generate one of these files for each region of the PTP plate, named `1.TCA.454Reads.qual`, etc. For the purposes of this tutorial, we use the quality scores file `Fasting_Example.qual`.

Mapping File (Tab-delimited `.txt`): The mapping file is generated by the user. This file contains all of the information about the samples necessary to perform the data analysis. At a minimum, the mapping file should contain the name of each sample, the barcode sequence used for each sample, the linker/primer sequence used to amplify the sample, and a Description column. In general, you should also include in the mapping file any metadata that relates to the samples (for instance, health status, or sampling site) and any additional information relating to specific samples that may be useful to have at hand when considering outliers (for example, what medications a patient was taking at time of sampling). Of note: the sample names may only contain alphanumeric characters (`A-z, 0-9`) and the dot (.). Full format specifications can be found in the Documentation (File Formats).

The Mapping file here is named `Fasting_Map.txt`. The contents of the mapping file are shown in Figure 1E.5.2. A nucleotide barcode sequence is provided for each of the nine samples, as well as metadata related to treatment group and date of birth, and general run descriptions about the project.

4. Check the mapping file.

Before beginning with QIIME, you should ensure that your mapping file is formatted correctly with the `check_id_map.py` script. Type:

```
check_id_map.py -m Fasting_Map.txt -o mapping_output
```

This module will display a message indicating whether or not problems were found in the mapping file. Errors and warnings will the output to a log file, which are present in the specified (`-o`) output directory. Errors will cause fatal problems with subsequent scripts and must be corrected before moving forward. Warnings will not cause fatal problems, but it is encouraged that you fix these problems, as they are often indicative of typos in your mapping file, invalid characters, or other unintended errors that will impact downstream analysis. A

corrected_mapping.txt file will also be created in the output directory, which will have a copy of the mapping file with invalid characters replaced, or a message indicating that no invalid characters were found.

5. Assign multiplexed reads to biological samples.

   The next task is to assign the multiplexed reads to samples based on their nucleotide barcode. Also, this step performs quality filtering based on the characteristics of each sequence, removing any low quality or ambiguous reads. The script for this step is split_libraries.py. A full description of parameters for this script are described in the Documentation. For this tutorial, we will use default parameters (minimum quality score = 25, minimum/ maximum length = 200/1000, error-correcting Golay of 12 nucleotide barcodes, no ambiguous base calls, and no mismatches allowed in the primer sequence).

Type:

```
split_libraries.py -m Fasting_Map.txt -f

Fasting_Example.fna -q Fasting_Example.qual -o

split_library_output
```

This invocation will create three files in the new directory split_library_output/:

split_library_log.txt: This file contains the summary of splitting, including the number of reads detected for each sample and a brief summary of any reads that were removed due to quality considerations.

histograms.txt: This tab-delimited file shows the number of reads at regular size intervals before and after splitting the library.

seqs.fna: This is a FASTA-formatted file where each sequence is renamed according to the sample it came from. The header line also contains the name of the read in the input FASTA file and information on any barcode errors that were corrected.

A few lines from the seqs.fna file are shown below:

```
>PC.634_1 FLP3FBN01ELBSX orig_bc=ACAGAGTCGGCT
new_bc=ACAGAGTCGGCT bc_diffs=0

CTGGGCCGTGTCTCAGTCCC…

>PC.634_2 FLP3FBN01EG8AX orig_bc=ACAGAGTCGGCT
new_bc=ACAGAGTCGGCT bc_diffs=0

TTGGACCGTGTCTCAGTTCCAATGT….

>PC.354_3 FLP3FBN01EEWKD orig_bc=AGCACGAGCCTA
new_bc=AGCACGAGCCTA bc_diffs=0

TTGGGCCGTGTCTCA…
```

Basic Protocol 2

Picking OTUs, Assigning Taxonomy, Inferring Phylogeny, and Creating an OTU Table

Basic Protocol 2 consists of picking Operational Taxonomic Units (OTUs) based on sequence similarity within the reads, and picking a representative sequence from each OTU. The protocol also assigns taxonomic identities using reference databases, aligns the OTU sequences, creates a phylogenetic tree, and constructs an OTU table, representing the abundance of each OTU in each microbial sample. Basic Protocol 2 requires demultiplexed sequences such as those generated in the seqs.fna file from Basic Protocol 1.

Necessary Resources—A functional installation of the QIIME VirtualBox is required; see Support Protocol for hardware requirements of the QIIME VirtualBox

1. Run the pick_otus_through_otu_table.py workflow, which performs a series of small steps by calling a series of other scripts automatically.

   This workflow consists of the following stages:

   a. Picking OTUs (for more information, refer to pick_otus.py).

   b. Picking a representative sequence set, one sequence from each OTU (for more information, refer to pick_rep_set.py).

   c. Assigning taxonomy to the representative sequence set (for more information, refer to assign_taxonomy.py).

   d. Aligning the representative sequence set (for more information, refer to align_seqs.py).

   e. Filtering the alignment prior to tree building and removing positions which are all gaps, or not useful for phylogenetic inference (for more information, refer to filter_alignment.py).

   f. Building a phylogenetic tree (for more information, refer to make_phylogeny.py).

   g. Building an OTU table (for more information, refer to make_otu_table.py).

   Using the output from split_libraries.py (the seqs.fna file), run the following command:

   ```
   pick_otus_through_otu_table.py -i split_library_output/
   seqs.fna -o otus
   ```

   The results of pick_otus_through_otu_table.py are in otus/, and a description of the steps performed and the results follow.

2. Inspect the results of taxonomy assignment.

   QIIME has performed a series of analysis stages, following the pick_otus_through_otu_table.py command from step 1. In this step we

inspect the results of stage (c), but first describe stages (a) through (c), as defined under step 1.

At stage (a), all of the sequences from all of the samples are clustered into Operational Taxonomic Units (OTUs) based on their sequence similarity. OTUs in QIIME are clusters of sequences, frequently intended to represent some degree of taxonomic relatedness. For example, when sequences are clustered at 97% sequence similarity with uclust, each resulting cluster is typically thought of as representing a species. This model and the current techniques for picking OTUs are inherently limited, however, in that 97% of OTUs do not match what humans have called species for many microbes. Determining exactly how OTUs should be defined, and what they represent, is an active area of research.

`pick_otus_through_otu_table.py` assigns sequences to OTUs at 97% similarity by default. Further information on how to view and change default behavior is discussed later.

Since each OTU may be made up of many related sequences, at stage (b) QIIME picks a representative sequence from each OTU for downstream analysis. This representative sequence will be used for taxonomic identification of the OTU and phylogenetic alignment. QIIME uses the OTU file created above and extracts a representative sequence from the FASTA file by one of several methods.

In the `otus/rep_set/` directory, QIIME has created two new files—the log file `seqs_rep_set.log` and the FASTA file `seqs_rep_set.fasta` containing one representative sequence for each OTU. In this FASTA file, the sequence has been renamed by the OTU, and the additional information on the header line reflects the sequence used as the representative:

```
>0 PC.636_424 CTGGGCCGTATCTCAGTCCCAATGTGGCCGGTCGACCTCTC….
```

```
>1 PC.481_321 TTGGGCCGTGTCTCAGTCCCAATGTGGCCGTCCGCCCTCTC….
```

A primary goal of the QIIME pipeline is to assign high-throughput sequencing reads to taxonomic identities using established databases. Stage (c) provides information on the microbial lineages found in microbial samples. By default, QIIME uses the Ribosomal Database Project (RDP) classifier to assign taxonomic data to each representative sequence from stage (b).

In the directory `otus/rdp_assigned_taxonomy/`, there will be a log file and a text file. The text file contains a line for each OTU considered, with the RDP taxonomy assignment and a numerical confidence of that assignment (1 is the highest possible confidence). For some OTUs, the assignment will be as specific as a bacterial species, while others may be assignable to nothing more specific than the bacterial domain.

Inspect the first few lines of the taxonomy assignment file by entering the command:

```
head otus/rdp_assigned_taxonomy/
seqs_rep_set_tax_assignments.txt
```

The first few lines of the text file should resemble those shown in Figure 1E.5.3.

3. Inspect the phylogenetic tree.

To infer the phylogenetic relationships relating the sequences, QIIME aligns the sequences in stage (d). Alignments can either be generated de novo using programs such as MUSCLE (Edgar, 2004), or through assignment to an existing alignment with tools like PyNAST (Caporaso et al., 2010). For small studies such as this tutorial, either method is possible. However, for studies involving many sequences (roughly, more than 1000), the de novo aligners are very slow and alignment with PyNAST is preferred. Since this is one of the most computationally intensive bottlenecks in the pipeline, large studies benefit greatly from parallelization of this task (described later). When using PyNAST as an aligner (the default), QIIME must know the location of a template alignment. Most QIIME installations use the greengenes file `core_set_aligned.fasta.imputed` by default.

After aligning the sequences, a log file and an alignment file are created in the directory `otus/pynast_aligned_seqs/`.

Before inferring a phylogenetic tree relating the sequences, it is beneficial to filter the sequence alignment to remove columns composed of only gaps and locations known to be excessively variable. Most QIIME installations use a lanemask file named either `lanemask_in_1s_and_0s.txt` or `lanemask_in_1s_and_0s` by default. Filtering is stage (e), and after filtering, a filtered alignment file is created in the directory `otus/pynast_aligned_seqs/`.

In stage (f), the filtered alignment file produced in the directory `otus/pynast_aligned_seqs/` is then used to build a phylogenetic tree using a tree-building program.

The Newick format tree file is written to `rep_set.tre`, which is located in the `otus/` directory. This file can be viewed in a tree visualization software, and is necessary for UniFrac diversity measurements and other phylogenetically aware analyses (described below).

To view the Newick-formatted tree as text, type:

```
less otus/rep_set.tre
```

type `q` when finished.

The tree obtained can also be visualized with programs such as FigTree (http://tree.bio.ed.ac.uk/software/figtree/), which was used to visualize the phylogenetic tree obtained from `rep_set.tre` in Figure 1E.5.4.

4. View statistics of the OTU table.

Using taxonomic assignments (stage c) and the OTU map (stage a), QIIME assembles a readable matrix of OTU abundance in each sample with meaningful taxonomic identifiers for each OTU.

The result of this step is `otu_table.txt`, which is located in the `otus/` directory. The first few lines of `otu_table.txt` are shown below (OTUs 1 to 9), where the first column contains the OTU number, the last column contains the taxonomic assignment for the OTU, and 9 columns between are for each of our 9 samples. The value of each `i`, `j` entry in the matrix is the number of times OTU `i` was found in the sequences for sample `j`.

To view the number of sequence reads that were assigned to each biological sample in the OTU table (`otus/otu_table.txt`), type:

```
per_library_stats.py -i otus/otu_table.txt
```

The output shows that there are relatively few sequences in this tutorial example, but the sequences present are fairly evenly distributed among the 9 microbial communities. The output is shown below:

Num samples: 9

Seqs/sample summary: Min: 146

Max: 150

Median: 148.0

Mean: 148.111111111

Std. dev.: 1.4487116456

Median Absolute Deviation: 1.0

Default even sampling depth in core_qiime_analyses.py (just a suggestion): 146

Seqs/sample detail:

PC.355: 146

PC.481: 146

PC.636: 147

PC.354: 148

PC.635: 148

PC.593: 149

PC.607: 149

PC.356: 150

PC.634: 150.

5. View a heatmap of the OTU table.

The QIIME pipeline includes a useful utility to generate images of the OTU table. The script is `make_otu_heatmap_html.py`. Type:

```
make_otu_heatmap_html.py -i otus/otu_table.txt -o otus/
OTU_Heatmap/
```

An html file is created in the directory `otus/OTU_Heatmap/`. In the menu bar at the top of the VirtualBox window, select Places: Home Folder, navigate to `qiime_totorial-v1.3.0/otus/OTU_Heatmap`, and double-click `otu_table.html`. You will be prompted to enter a value for "Filter by Counts per OTU." Leave the filter value unchanged, click the Sample ID button, and a graphic will be generated. For each sample, you will see in a heatmap the number of times each OTU was found in that sample. Mouse over any individual count to get more information on the OTU (including taxonomic assignment). Within the mouseover, there is a link for the terminal lineage assignment, so you can easily search Google for more information about that assignment (see Fig. 1E.5.5).

Only OTUs with total counts at or above the threshold specified by Filter by counts OTU will be displayed. The OTU heatmap displays raw OTU counts per sample, where the counts are colored based on the contribution of each OTU to the total OTU count present in that sample (blue: contributes low percentage of OTUs to sample; red: contributes high percentage of OTUs).

Alternatively, you can click on one of the counts in the heatmap and a new pop-up window will appear. The pop-up window uses a Google Visualization API called Magic-Table. Depending on which table count you clicked on, the pop-up window will put the clicked-on count in the middle of the pop-up heatmap (Fig. 1E.5.6).

On the original heatmap Web page, click the Taxonomy button: you will generate a heatmap keyed by taxon assignment, which allows you to conveniently look for organisms and lineages of interest in your study. Again, mousing over an individual count will show additional information for that OTU and sample (Fig. 1E.5.7).

6. View taxonomic summary information for each community. Next, group OTUs by different taxonomic levels (division, class, family, etc.) with the workflow script `summarize_taxa_through_plots.py`. Note that this process depends directly on the method used to assign taxonomic information to OTUS (see Assign Taxonomy above). In the (black) terminal, type:

```
summarize_taxa_through_plots.py -i otus/otu_table.txt -o
wf_taxa_summary -m Fasting_Map.txt
```

The script will generate a new table grouping sequences by taxonomic assignment at various levels, for example the phylum level table at `wf_taxa_summary/Taxa_Charts/otu_table_L3.txt`.

The value of each `i`, `j` entry in the matrix is the count of the number of times all OTUs belonging to the taxon `i` (for example, Phylum Actinobacteria) were found in the sequences for sample `j`.

To view the resulting charts, again go to Places: Home Folder in the menu bar, navigate to `qiime_tutorial-v1.3.0/wf_taxa_summary`, and open the `area_charts` html file located in the `taxa_summary_plots/` folder. The area chart (Fig. 1E.5.8) shows the taxa assignments for each sample as an area chart; the same information is available as a bar chart (Fig. 1E.5.9). Mouse over the plot to see which taxa are contributing to the percentage shown.

Basic Protocol 3

Alpha Diversity within Samples and Rarefaction Curves

Basic Protocol 3 consists of computing the within community diversity (alpha diversity) for each of the 9 microbial communities, and generating rarefaction curves (graphs of diversity versus sequencing depth). Basic Protocol 3 requires an OTU table and phylogenetic tree such as those produced in Basic Protocol 2.

Necessary Resources—A functional installation of the QIIME VirtualBox is required; see Support Protocol for hardware requirements of the QIIME VirtualBox.

1. Run the alpha diversity workflow.

   Community ecologists use a variety of techniques to describe the microbial diversity within their study. This diversity can be assessed within a community (alpha diversity) or between a collection of samples (beta diversity). Here, we will determine the level of alpha diversity in our samples with QIIME. To perform this analysis, we will use the `alpha_rarefaction.py` workflow script. This script performs the following stages:

   a. Generate rarefied OTU tables (for more information, refer to `multiple_rarefactions.py`).

   b. Compute measures of alpha diversity for each rarefied OTU table (for more information, refer to `alpha_diversity.py`).

   c. Collate alpha diversity results (for more information, refer to `collate_alpha.py`).

   d. Generate alpha diversity rarefaction plots (for more information, refer to `make_rarefaction_plots.py`).

   Although we could run this workflow with the (sensible) default parameters, this provides an excellent opportunity to illustrate the use of custom parameters.

To see what measures of alpha diversity will be computed by default, type:

```
alpha_diversity.py -h
```

You should see, among other information:

```
-m METRICS, --metrics=METRICS
```

This represents the alpha-diversity metric(s) to use. A comma-separated list should be provided when multiple metrics are specified. [default: `PD_whole_tree, chao1, observed_species`].

To also use the shannon index (an alpha diversity measure derived from information theory), create a custom parameters file by typing:

```
echo "alpha_diversity:metrics shannon, PD_whole_tree, chao1,
observed_species" > alpha_params.txt
```

Then run the workflow, which requires the OTU table (`-i`) and phylogenetic tree (`-t`) from the section above, and the custom parameters file we just created:

```
alpha_rarefaction.py -i otus/otu_table.txt -m
Fasting_Map.txt -p alpha_params.txt -t otus/rep_set.tre -o
wf_arare/
```

Descriptions of the steps involved in `alpha_rarefaction.py` follow:

The directory `wf_arare/rarefaction/` will contain many text files named `rarefaction_##_#.txt`; the first set of numbers represents the number of sequences sampled, and the last number represents the iteration number. If you opened one of these files—created in stage (a)—you would find an OTU table where for each sample the sum of the counts equals the number of samples taken.

The rarefaction tables are the basis for calculating diversity metrics, which reflect the diversity within the sample based on the abundance of various taxa within a community. The QIIME pipeline allows users to conveniently calculate more than two dozen different diversity metrics. The full list of available metrics is available here. Every metric has different strengths and limitations—technical discussion of each metric is readily available online and in ecology textbooks, but is beyond the scope of this document. By default, QIIME calculates three metrics:

  a. Chao1 metric estimates the species richness

  b. The Observed Species metric is simply the count of unique OTUs found in the sample.

  c. Phylogenetic Distance (PD_whole_tree) is the only phylogenetic metric used. It requires a phylogenetic tree, which is frequently generated earlier in the analysis (see Basic Protocol 2).

  d. In addition, `alpha_params.txt` specified above adds the shannon index to the list of alpha diversity measures calculated by QIIME. The shannon index is the information entropy of the observed OTU abundances, and accounts for both richness and evenness.

The result of stage (ii) produces several text files with the results of the alpha diversity computations performed on the rarefied OTU tables. The results are located in the `wf_arare/alpha_div/` directory.

The output directory `wf_arare/alpha_div/` contains one text file `alpha_rarefaction_##_#` for every file input from `wf_arare/rarefaction/`, where the numbers represent the number of samples and iterations as before. The content of this tab delimited file is the calculated metrics for each sample. To collapse the individual files into a single combined table, the workflow [in stage (iii)] used the script `collate_alpha.py`.

In the newly created directory `wf_arare/alpha_div_collated/`, there will be one matrix for every alpha diversity metric used. This matrix will contain the metric for every sample, arranged in ascending order from lowest number of sequences per sample to highest.

QIIME creates plots of alpha diversity versus simulated sequencing effort, known as rarefaction plots, using the script `make_rarefaction_plots.py`, in stage (iv). This script takes a mapping file and any number of rarefaction files generated by `collate_alpha.py` and creates rarefaction curves. Each curve represents a sample and can be colored by the sample metadata supplied in the mapping file.

This step generates a `wf_arare/alpha_rarefaction_plots/rarefaction_plots.html` that can be opened with a Web browser, in addition to other files. The `wf_arare/alpha_rarefaction_plots/average plots/` folder contains the average plots for each metric and category and the `alpha_rarefaction_plots/html_plots/` folder contains all the images used in the html page generated.

2. View the rarefaction plots. Open `wf_arare/alpha_rarefaction_plots/rarefaction_plots.html` in a Web browser by double-clicking on it. Once the browser window is open, select the metric `PD_whole_tree` and the category Treatment, to reveal a plot like Figure 1E.5.10. Turn on/off lines in the plot by (un)checking the box next to each label in the legend, and experiment with clicking on the triangle next to each label in the legend to see all the samples that contribute to that category.

Below each plot is a table displaying average values for each measure of alpha diversity for each group of samples the specified category.

Basic Protocol 4

Eta Diversity Between Samples and Beta Diversity Plots

Basic Protocol 4 consists of computing the between community diversity (beta diversity) for each of the nine microbial communities, and generating Principal Coordinates Analysis (PCoA) plots and distance histograms representing the relationships among the nine microbial communities (for background, see, e.g., Legendre and Legendre, 1998). Basic Protocol 4 requires an OTU table and phylogenetic tree such as those produced in Basic Protocol 2.

Necessary Resources—The OTU table and Phylogenetic tree produced in Basic Protocol 2, and the mapping file from Basic Protocol 1.

1. Run the beta diversity workflow.

    Beta diversity represents the explicit comparison of microbial (or other) communities based on their composition. Beta-diversity metrics thus assess the differences between microbial communities. The fundamental output of these comparisons is a square matrix where a "distance" or dissimilarity is calculated between every pair of community samples, reflecting the dissimilarity between those samples. The data in this distance matrix can be visualized with analyses such as Principal Coordinate Analysis (PCoA) and hierarchical clustering. Like alpha diversity, there are many possible metrics which can be calculated with the QIIME pipeline. Here, we will calculate beta diversity between our 9 microbial communities using the default beta diversity metrics of weighted and unweighted unifrac, which are phylogenetic measures used extensively in recent microbial community sequencing projects.

    For this analysis we use the script `jackknifed_beta_diversity.py`, which performs a series of analyses consisting of the following stages:

    a. Compute the beta diversity distance matrix from the full OTU table (and tree, if applicable) (for more information, refer to `beta_diversity.py`).

    b. Build UPGMA tree from full distance matrix; (for more information, refer to `upgma_cluster.py`).

    c. Build rarefied OTU tables (for more information, refer to `multiple_rarefactions.py`).

    d. Compute distance matrices for rarefied OTU tables (for more information, refer to `beta_diversity.py`).

    e. Build UPGMA trees from rarefied distance matrices (for more information, refer to `upgma_cluster.py`).

    f. Compare rarefied UPGMA trees and determine jackknife support for tree nodes (for more information, refer to `tree_compare.py` and `consensus_tree.py`).

    g. Compute principal coordinates on each rarefied distance matrix (for more information, refer to `principal_coordinates.py`).

    h. Compare rarefied principal coordinates plots from each rarefied distance matrix (for more information, refer to `make_3d_plots.py` and `make_2d_plots.py`).

    To run the analysis, type the following:

    ```
    jackknifed_beta_diversity.py -i otus/otu_table.txt -t otus/
    rep_set.tre -m Fasting_Map.txt -o wf_jack -e 110
    ```

2. Create a jackknife supported tree, and view the result.

Unweighted Pair Group Method with Arithmetic mean (UPGMA) is a type of hierarchical clustering method using average linkage and can be used to interpret the distance matrix produced by `beta_diversity.py`. Stages (a) and (b) produced a Newick-formatted tree relating the samples, at `wf_jack/unweighted_unifrac/otu_table_upgma.tre`.

To measure the robustness of this result to sequencing effort, QIIME performs a jackknifing analysis, wherein a smaller number of sequences are chosen at random from each sample, and the resulting UPGMA tree from this subset of data is compared with the tree representing the entire available data set. This process is repeated with many random subsets of data, and the tree nodes that prove more consistent across jackknifed datasets are deemed more robust [stages (c) through (f)].

First the jackknifed OTU tables were generated, by subsampling the full available data set. In this tutorial, each sample initially contained between 146 and 150 sequences. To ensure that a random subset of sequences is selected from each sample, select 110 sequences from each sample (75% of the smallest sample, though this value is only a guideline), which was designated by the `-e` option when running the workflow script above.

More jackknife replicates provide a better estimate of the variability expected in beta diversity results, but at the cost of longer computational time. By default, QIIME generates 10 jackknife replicates of the available data. Each replicate is a simulation of a smaller sequencing effort (110 sequences in each sample, as defined below).

The workflow then calculated the distance matrix for each jackknifed dataset, but now in batch mode, which resulted in two sets of 10 distance matrix files written to the `wf_jack/unweighted_unifrac/rare_dm/` and `wf_jack/weighted_unifrac/rare_dm/` directories. Each of those was then used as the basis for hierarchical clustering with UP-GMA, written to the `wf_jack/unweighted_unifrac/rare_upgma/` and `wf_jack/weighted_unifrac/rare_upgma/` directories.

UPGMA clustering of the 10 distance matrix files results in 10 hierarchical clusters of the 9 mouse microbial communities, with each hierarchical cluster based on a random sub-sample of the available sequence data.

This compares the UPGMA clustering based on all available data with the jack-knifed UPGMA results. Three files are written to `wf_jack/unweighted_unifrac/upgma_cmp/` and `wf_jack/weighted_unifrac/upgma_cmp/`:

`master_tree.tre`, which is virtually identical to `jackknife_named_nodes.tre` but each internal node of the UPGMA clustering is assigned a unique name

`jackknife_named_nodes.tre`

`jackknife_support.txt`, which explains how frequently a given internal node had the same set of descendant samples in the jackknifed UPGMA clusters as it does in the UPGMA cluster using the full available data. A value of 0.5 indicates that half of the jackknifed data sets support that node, while 1.0 indicates perfect support.

`jackknife_named_nodes.tre` can be viewed with FigTree or another tree-viewing program. However, as an example, we can visualize the bootstrapped tree using QIIME's `make_bootstrapped_tree.py`, as follows.

Type:

```
make_bootstrapped_tree.py -m wf_jack/unweighted_unifrac/
upgma_cmp/master_tree.tre -s wf_jack/unweighted_unifrac/
upgma_cmp/jackknife_support.txt -o wf_jack/
unweighted_unifrac/upgma_cmp/jackknife_named_nodes.pdf
```

Open the resulting pdf by typing:

```
gnome-open wf_jack/unweighted_unifrac/upgma_cmp/
jackknife_named_nodes.pdf
```

Figure 1E.5.11 shows the tree with internal nodes colored, red for 75% to 100% support, yellow for 50% to 75%, green for 25% to 50%, and blue for <25% support. Although PC.354 and PC.593 cluster together, we do not have high confidence in that result. However, there is excellent jackknife support for all fasted samples (`PC.6××`) clustered together, separate from the non-fasted samples.

3. Inspect the jackknife-supported PCoA plots.

    The jackknifed replicate PCoA plots created in stages (g) and (h) can be compared to assess the degree of variation from one replicate to the next. QIIME displays this variation by displaying confidence ellipsoids around the samples represented in a PCoA plot.

    Navigate to `wf_jack/unweighted_unifrac/3d_plots/` and open `pcoa_unweighted_unifrac_rarefaction_110_0_3D_PCoA_plots.html`.

    Scroll down the top right corner of the window that appears to select `Treatment_unscaled`.

    Example results are shown in Figure 1E.5.12. By default, the script will plot the first three dimensions in your file. Other combinations can be viewed using the "Views:Choose viewing axes" option in the KiNG viewer. The first 10 components can be viewed using "Views:Parallel coordinates" option or typing " /".

Support Protocol

Installing QIIME VIA VirtualBox

QIIME can be run in many environments, from a laptop running Windows to a high-performance computer cluster. It includes parallelization of many of the computationally complex steps, so if an analysis is taking unacceptably long, or requires more resources than are present on a single machine, it is likely worth investigating the use of QIIME on Amazon EC2, or installing QIIME natively on a compute cluster. In this protocol we discuss a simple way of installing QIIME via the VirtualBox for use on a single computer.

Necessary Resources—You will need a computer with a 64-bit processor and the capability of running a 64-bit operating system as a VirtualBox guest OS. Most modern personal computers running Windows, MacOS X, or Linux operating systems qualify. You will also need about 10 GB of free storage, and approximately 2 GB of memory.

1. Download and install the VirtualBox (VB) version for your machine from http://www.virtualbox.org/.

2. Download the 64-bit QIIME Virtual Box from http://bmf.colorado.edu/QIIME/QIIME-1.3.0-amd64.vdi.gz.

   This file is large (>1 GB) so it may take between a few minutes and a few hours depending on your Internet connection speed. You will need to unzip this file, which you can typically do by double-clicking on it.

3. Launch VirtualBox, and create a new machine (press the New button).

4. A new window will appear. Click Next or Continue.

5. In this screen type QIIME as the name for the virtual machine. Then select Linux as the Operating System, and Ubuntu (64 bit) as the version. Click Next or Continue.

6. Select the amount of RAM (memory). You will need at least 1024 MB, but the best option is based on your machine. If you are unsure, select 1024 MB. After selecting the amount of RAM, click Next or Continue.

7. Select "Use existing hard drive," and click the folder icon next to the selector (it has a green "up" arrow). In the new window click Add, and locate the virtual hard drive that was downloaded in step 2. Click Select and then click Next or Continue.

8. In the new window, click Finish.

9. Double-click on the new virtual machine created (it will be called QIIME) to boot it for the first time.

Commentary

Background Information

Sequence-based microbial ecology studies, which encompass whole metagenome shotgun metagenomics, metatranscriptomics, and amplicon (e.g., 16S rRNA) sequencing, are

increasingly prevalent and increasingly large in scale. The usefulness of a powerful analysis pipeline is thus apparent. However, given the rapid ongoing progression of sequencing technologies (Quail et al., 2008; Schwartz and Waterman, 2010), as well as the increase in our understanding of how microbial communities are structured and how they differ across habitats and times (Arumugam et al., 2011; Caporaso et al., 2011), the development of computational tools must keep pace with a continually and quickly changing set of objectives. Speaking to this, one of the key design decisions in the development of QIIME was the choice to use existing implementations of algorithms—tools such as FastTree for heuristic based maximum-likelihood phylogeny inference (Price et al., 2010), the RDP classifier for the assignment of taxonomic data using a naïve bayesian classifier (Wang et al., 2007), and others. This allows QIIME, which continues to undergo development, to easily and relatively quickly adapt novel stand-alone tools, and thus improve in step with advances in the field of microbial community ecology.

QIIME includes broad workflow scripts to abstract out some of the complexity of the analysis of microbial sequence analysis. QIIME scripts have sensible default values for most parameters of interest, thus allowing users to obtain reasonable results without requiring detailed decision making at each step of the (typically) long analysis process. However, researchers with unique needs, or preferences for alternative approaches (e.g., different measures of between community beta-diversity, or different reference databases for taxonomic identification), are easily able to customize the behavior of QIIME, simply by modifying settings away from their default values. QIIME, while a powerful and by necessity somewhat complex analysis pipeline, also performs straightforward analyses with a minimum of user intervention, while making clear the default protocols that have been performed.

Critical Parameters

The data used in these protocols is intentionally limited, to allow for faster execution times. However, many analyses will contain significantly more sequences, and thus will benefit significantly from the parallelization of many analysis steps.

Most of these steps can be run using the workflow scripts, some of which were mentioned above. To run the workflow scripts in parallel, pass the `-a` option to each of the scripts, and optionally the `-O` option to specify the number of parallel jobs to start. If running on a quad-core computer, one can set the number of jobs to start as 4 for one of the workflow scripts as follows:

```
pick_otus_through_otu_table.py -i split_library_output/seqs.fna -
o otus -a -O 4
```

Troubleshooting

All scripts it QIIME have built-in help, accessible by typing `script_name.py -h`. In addition, documentation exists at http://www.qiime.org, and the help forum at http://forum.qiime.org is typically quite active and useful.

Acknowledgments

Literature Cited

Arumugam M, Raes J, Pelletier E, Le Paslier D, Yamada T, Mende DR, Fernandes GR, Tap J, Bruls T, Batto JM, Bertalan M, Borruel N, Casellas F, Fernandez L, Gautier L, Hansen T, Hattori M, Hayashi T, Kleerebezem M, Kurokawa K, Leclerc M, Levenez F, Manichanh C, Nielsen HB, Nielsen T, Pons N, Poulain J, Qin J, Sicheritz-Ponten T, Tims S, Torrents D, Ugarte E, Zoetendal EG, Wang J, Guarner F, Pedersen O, de Vos WM, Brunak S, Doré J, Antolín M, Artiguenave F, Blottiere HM, Almeida M, Brechot C, Cara C, Chervaux C, Cultrone A, Delorme C, Denariaz G, Dervyn R, Foerstner KU, Friss C, van de Guchte M, Guedon E, Haimet F, Huber W, van Hylckama-Vlieg J, Jamet A, Juste C, Kaci G, Knol J, Lakhdari O, Layec S, Le Roux K, Maguin E, Mérieux A, Melo Minardi R, M'rini C, Muller J, Oozeer R, Parkhill J, Renault P, Rescigno M, Sanchez N, Sunagawa S, Torrejon A, Turner K, Vandemeulebrouck G, Varela E, Winogradsky Y, Zeller G, Weissenbach J, Ehrlich SD, Bork P. MetaHIT Consortium. Enterotypes of the human gut microbiome. Nature. 2011; 474:666.

Caporaso JG, Bittinger K, Bushman FD, De-Santis TZ, Andersen GL, Knight R. PyNAST: A flexible tool for aligning sequences to a template alignment. Bioinformatics. 2010; 26:266–267. [PubMed: 19914921]

Caporaso JG, Lauber CL, Walters WA, Berg-Lyons D, Lozupone CA, Turnbaugh PJ, Fierer N, Knight R. Global patterns of 16s rrna diversity at a depth of millions of sequences per sample. Proc Natl Acad Sci U S A. 2011; 108:4516. [PubMed: 20534432]

Crawford PA, Crowley JR, Sambandam N, Muegge BD, Costello EK, Hamady M, Knight R, Gordon JI. Regulation of myocardial ketone body metabolism by the gut microbiota during nutrient deprivation. Proc Natl Acad Sci U S A. 2009; 106:11276–11281. [PubMed: 19549860]

Edgar RC. MUSCLE: A multiple sequence alignment method with reduced time and space complexity. BMC Bioinformatics. 2004; 5:113. [PubMed: 15318951]

Legendre, P.; Legendre, L. Numerical Ecology. Elsevier Science; New York: 1998.

Price MN, Dehal PS, Arkin AP. Fasttree 2-approximately maximum-likelihood trees for large alignments. PLoS One. 2010; 5:e9490. [PubMed: 20224823]

Quail MA, Kozarewa I, Smith FA, Scally P, Stephens J, Durbin R, Swerdlow H, Turner DJ. A large genome center's improvements to the illumina sequencing system. Nat Methods. 2008; 5:1005–1010. [PubMed: 19034268]

Schwartz DC, Waterman MS. New generations: Sequencing machines and their computational challenges. J Comp Sci Tech. 2010; 25:3–9.

Wang Q, Garrity GM, Tiedje JM, Cole JR. Naive bayesian classifier for rapid assignment of rrna sequences into the new bacterial taxonomy. Appl Environ Microbiol. 2007; 73:5261. [PubMed: 17586664]
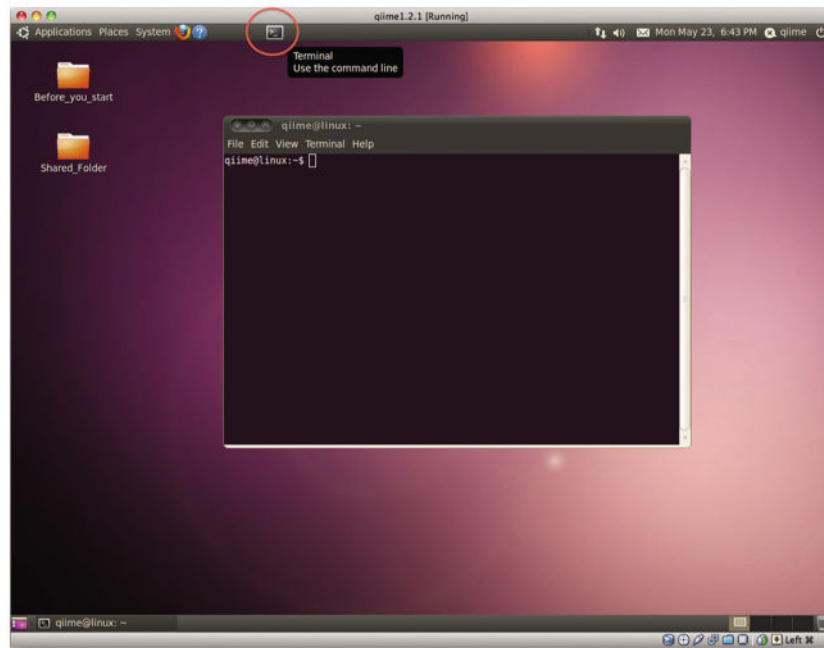
Figure 1E.5.1.
A screenshot of the QIIME VirtualBox, with the terminal icon indicated, and a terminal window open.

#SampleID	BarcodeSequence	LinkerPrimerSequence	Treatment	DOB	Description
#Example mapping file for the QIIME analysis package.					
PC.354	AGCACGAGCCTA	YATGCTGCCTCCCGTAGGAGT	Control	20061218	Control_mouse_I.D._354
PC.355	AACTCGTCGATG	YATGCTGCCTCCCGTAGGAGT	Control	20061218	Control_mouse_I.D._355
PC.356	ACAGACCACTCA	YATGCTGCCTCCCGTAGGAGT	Control	20061126	Control_mouse_I.D._356
PC.481	ACCAGCGACTAG	YATGCTGCCTCCCGTAGGAGT	Control	20070314	Control_mouse_I.D._481
PC.593	AGCAGCACTTGT	YATGCTGCCTCCCGTAGGAGT	Control	20071210	Control_mouse_I.D._593
PC.607	AACTGTGCGTAC	YATGCTGCCTCCCGTAGGAGT	Fast	20071112	Fasting_mouse_I.D._607
PC.634	ACAGAGTCGGCT	YATGCTGCCTCCCGTAGGAGT	Fast	20080116	Fasting_mouse_I.D._634
PC.635	ACCGCAGAGTCA	YATGCTGCCTCCCGTAGGAGT	Fast	20080116	Fasting_mouse_I.D._635
PC.636	ACGGTGAGTGTC	YATGCTGCCTCCCGTAGGAGT	Fast	20080116	Fasting_mouse_I.D._636

Figure 1E.5.2.

Contents of the mapping file (`Fasting_Map.txt`). Note that the SampleIDs contain only letters, numbers, and period characters.

```
358 PC.607_595  Bacteria;Firmicutes;Clostridia;Clostridiales;Incertae Sedis XIII    0.920
327 PC.607_1153 Bacteria    1.000
98 PC.356_336   Bacteria;Firmicutes;Clostridia;Clostridiales;Lachnospiraceae    0.810
266 PC.634_25   Bacteria;Firmicutes;Clostridia;Clostridiales;Lachnospiraceae    0.860
126 PC.354_692  Bacteria;Firmicutes;Clostridia;Clostridiales;Lachnospiraceae    0.830
```

Figure 1E.5.3.

The first few lines of the taxonomy assignment file, showing on each line the OTU identifier, the representative sequence identifier, the taxonomy assigned to that sequence, and the confidence in that assignment.

Figure 1E.5.4.
A visualization of the phylogenetic tree using FigTree. The tips are unlabeled here, but can be inspected interactively.
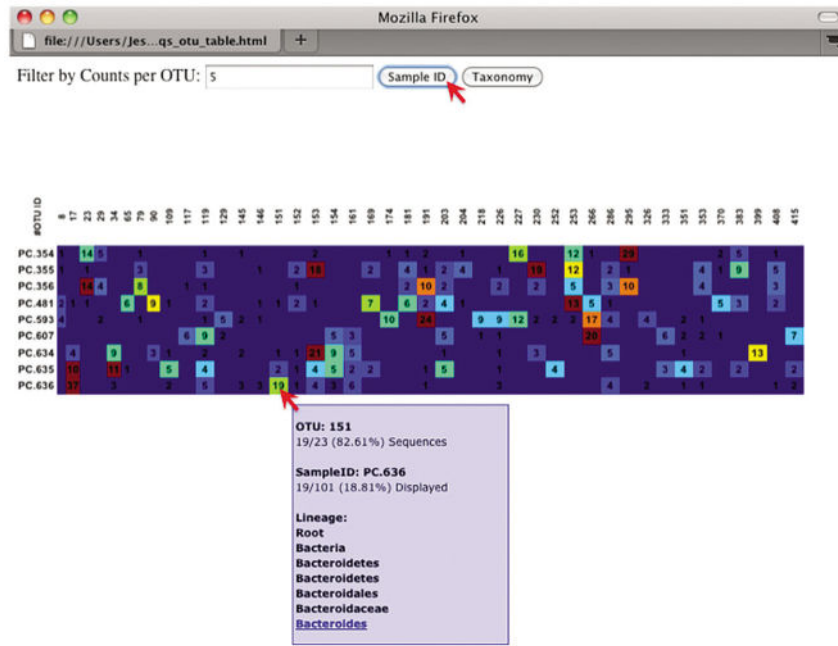
Figure 1E.5.5.
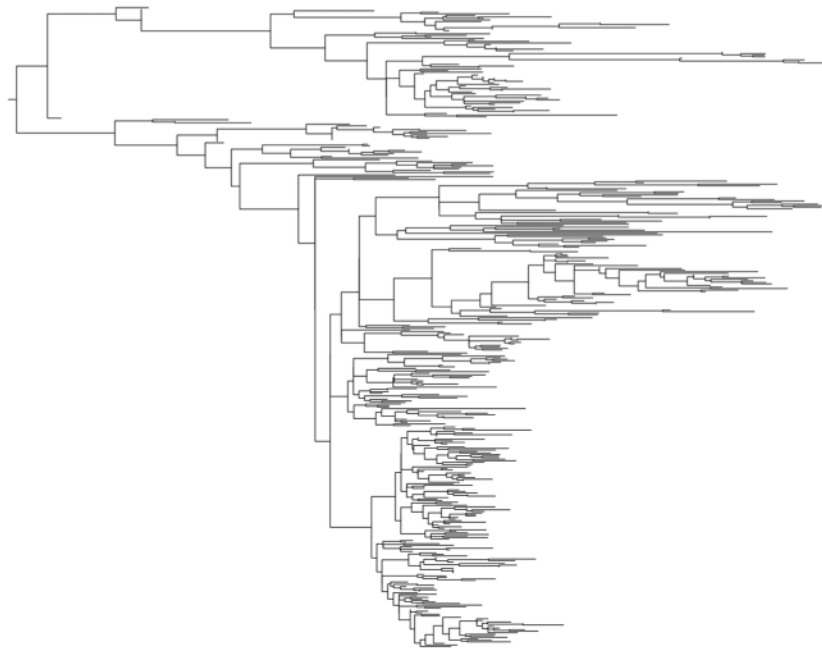An OTU table heatmap, showing the relative abundance of each OTU within each microbial community.
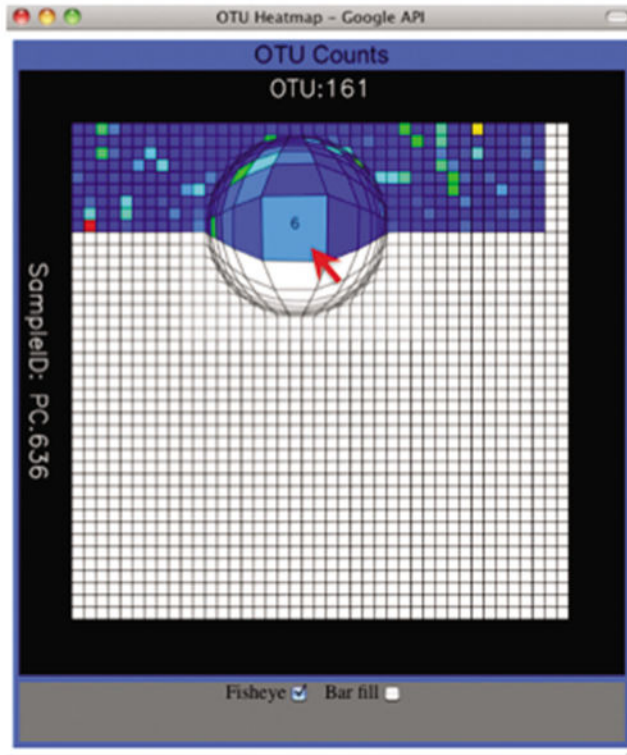
Figure 1E.5.6.
Magic-Table visualization of the OTU table heatmap.

Figure 1E.5.7.
An OTU table heatmap showing taxonomy assignment for each OTU.

Taxonomy Summary. Current Level: Phylum
View Figure (.pdf)   View Legend (.pdf)

Legend:
- Root.Bacteria.Actinobacteria
- Root.Bacteria.Bacteroidetes
- Root.Bacteria.Deferribacteres
- Root.Bacteria.Firmicutes
- Root.Bacteria.Other
- Root.Bacteria.Proteobacteria
- Root.Bacteria.TM7
- Root.Bacteria.Verrucomicrobia

View Table (.txt)

Legend	Taxonomy	Total count	Total %	PC.354 count	PC.354 %	PC.355 count	PC.355 %	PC.356 count	PC.356 %	PC.481 count	PC.481 %	PC.593 count	PC.593 %	PC.607 count	PC.607 %	PC.634 count	PC.634 %	PC.635 count	PC.635 %	PC.636 count	PC.636 %
	Root;Bacteria;Actinobacteria	0	0.60	0	0.00%	0	0.00%	0	0.00%	0	0.68%	0	0.00%	0	1.34%	0	2.00%	0	0.68%	0	0.68%
	Root;Bacteria;Bacteroidetes	3	31.27	0	4.73%	0	27.40%	0	10.67%	0	11.64%	0	21.48%	0	27.52%	1	64.67%	0	46.62%	1	66.67%
	Root;Bacteria;Deferribacteres	0	1.27	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	2.01%	0	3.33%	0	1.35%	0	4.76%
	Root;Bacteria;Firmicutes	5	57.59	1	92.57%	1	69.86%	1	76.00%	1	82.88%	1	55.03%	1	57.05%	0	24.67%	0	38.51%	0	21.77%
	Root;Bacteria;Other	1	8.23	0	2.70%	0	2.74%	0	13.33%	0	4.79%	0	20.13%	0	10.07%	0	2.00%	0	12.84%	0	5.44%
	Root;Bacteria;Proteobacteria	0	0.82	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	3.36%	0	2.01%	0	1.33%	0	0.00%	0	0.68%
	Root;Bacteria;TM7	0	0.15	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	1.33%	0	0.00%	0	0.00%
	Root;Bacteria;Verrucomicrobia	0	0.07	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.67%	0	0.00%	0	0.00%

NOTE: the counts displayed pertain to either relative or absolute values depending on your selection from summarize_taxa.py. For relative values, the numbers are converted to integer, so counts below 0.5 appear as 0.

Figure 1E.5.8.
An area chart showing the relative abundance of each phylum within each microbial community.

Taxonomy Summary. Current Level: Phylum
View Figure (.pdf) View Legend (.pdf)

Legend:
- Root.Bacteria.Actinobacteria
- Root.Bacteria.Bacteroidetes
- Root.Bacteria.Deferribacteres
- Root.Bacteria.Firmicutes
- Root.Bacteria.Other
- Root.Bacteria.Proteobacteria
- Root.Bacteria.TM7
- Root.Bacteria.Verrucomicrobia

View Table (.txt)

Legend	Taxonomy	Total count	Total %	PC.354 count	PC.354 %	PC.355 count	PC.355 %	PC.356 count	PC.356 %	PC.481 count	PC.481 %	PC.593 count	PC.593 %	PC.607 count	PC.607 %	PC.634 count	PC.634 %	PC.635 count	PC.635 %	PC.636 count	PC.636 %
	Root;Bacteria;Actinobacteria	0	0.60	0	0.00%	0	0.00%	0	0.00%	0	0.68%	0	0.00%	0	1.34%	0	2.00%	0	0.68%	0	0.68%
	Root;Bacteria;Bacteroidetes	3	31.27	0	4.73%	0	27.40%	0	10.67%	0	11.64%	0	21.48%	0	27.52%	1	64.67%	0	46.62%	1	66.67%
	Root;Bacteria;Deferribacteres	0	1.27	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	2.01%	0	3.33%	0	1.35%	0	4.76%
	Root;Bacteria;Firmicutes	5	57.59	1	92.57%	1	69.86%	1	76.00%	1	82.88%	1	55.03%	1	57.05%	0	24.67%	0	38.51%	0	21.77%
	Root;Bacteria;Other	1	8.23	0	2.70%	0	2.74%	0	13.33%	0	4.79%	0	20.13%	0	10.07%	0	2.00%	0	12.84%	0	5.44%
	Root;Bacteria;Proteobacteria	0	0.82	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	3.36%	0	2.01%	0	1.33%	0	0.00%	0	0.68%
	Root;Bacteria;TM7	0	0.15	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	1.33%	0	0.00%	0	0.00%
	Root;Bacteria;Verrucomicrobia	0	0.07	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.00%	0	0.67%	0	0.00%	0	0.00%

NOTE: the counts displayed pertain to either relative or absolute values depending on your selection from summarize_taxa.py. For relative values, the numbers are converted to integer, so counts below 0.5 appear as 0.

Figure 1E.5.9.
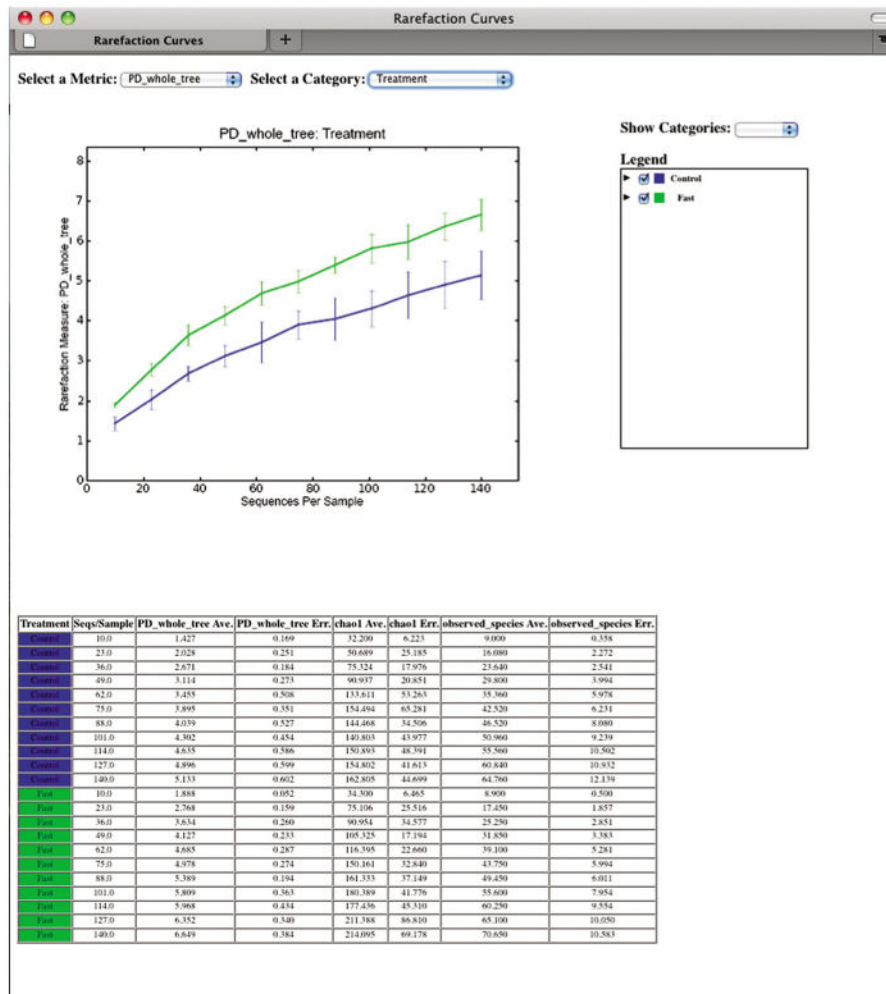A bar chart of phylum level abundance within communities, similar to Figure 10.7.8.

Figure 1E.5.10.

A Web browser window displaying rarefaction plots. The vertical axis displays the diversity of the community, while the horizontal axis displays the number of sequences considered in the diversity calculation. Each line on the figure represents the average of all microbial belonging to a group within a category: here the green line represents all fasted mouse communities, and the blue line represents the control communities.
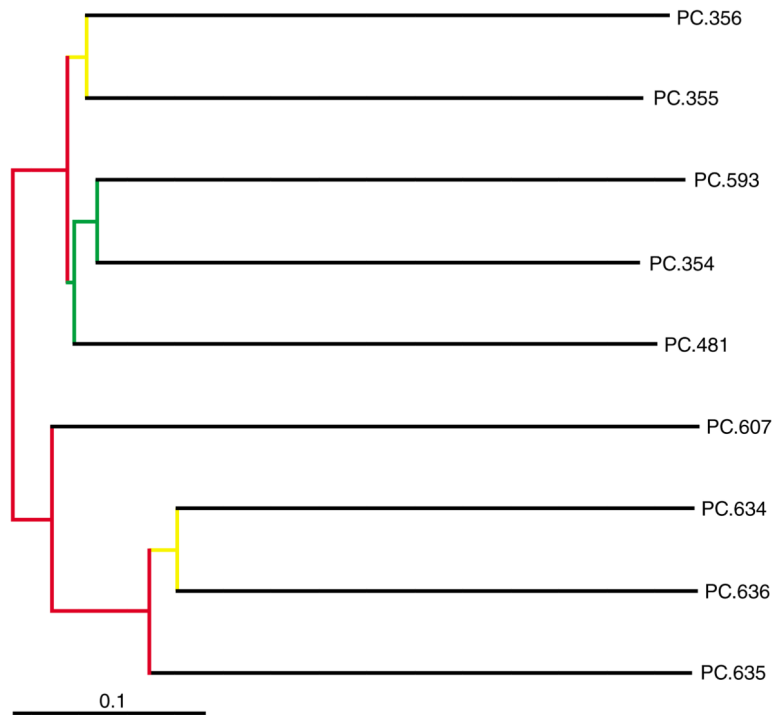
Figure 1E.5.11.
A visualization of bootstrap-supported hierarchical clustering of the 9 microbial communities under investigation. Note that the fasted mouse communities (`PC.6××`) cluster together, and the result is supported by jackknife tests (red implies > 75% support).
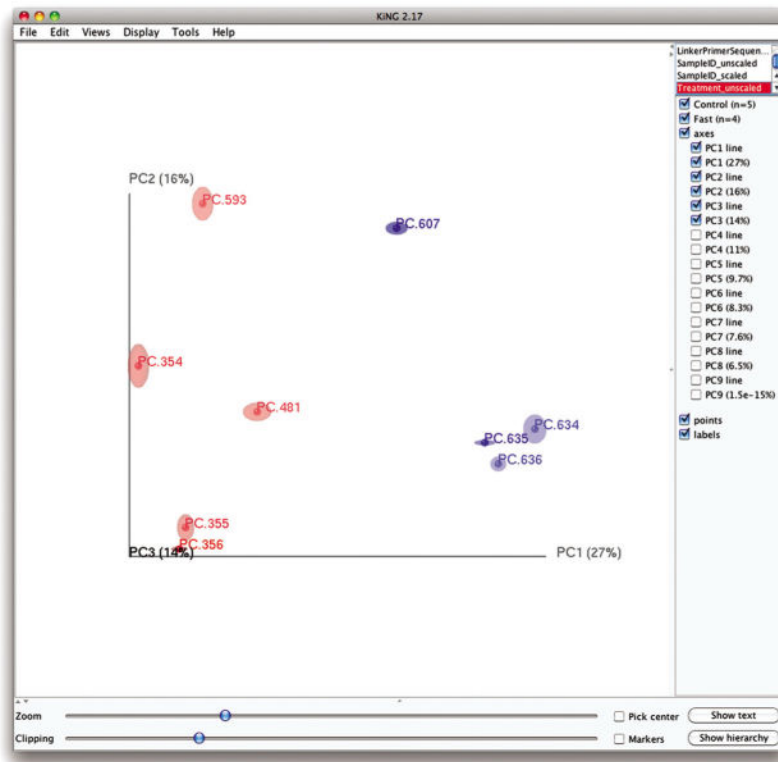
Figure 1E.5.12.
A Principal Coordinates plot of the 9 communities, showing jackknife-supported confidence ellipsoids. The first two principal axes are shown.